Compression in Language Modeling

Team: MIT-Han-Lab

Demi Guo*[2], Hanrui Wang*[1], Zhongxia Yan*[1], Phillip Isola[1] and Song Han[1]

[1]Massachusetts Institute of Technology
[2]Harvard University

*Equal contribution, alphabetical order

Observations

- Desirable model properties
  - Predictive (< 35 PPL)
  - Fast to train
  - Efficient inference
- **Score = param/159M + compute/318M**
  - No penalty for non-parametric memory
  - Rely more on memory!

LSTM

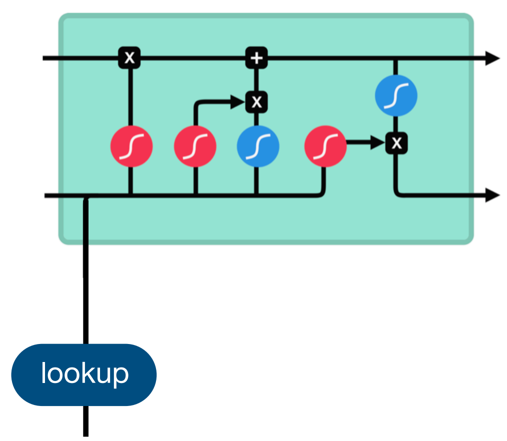
- Sequential in time, slow to train
- Harder to learn long-term dependencies

This **is** **a** **sentence**

Image from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM

- Sequential in time, slow to train
- Harder to learn long-term dependencies



Word Embed

This **is** **a** **sentence**

Image from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM

- Sequential in time, slow to train
- Harder to learn long-term dependencies

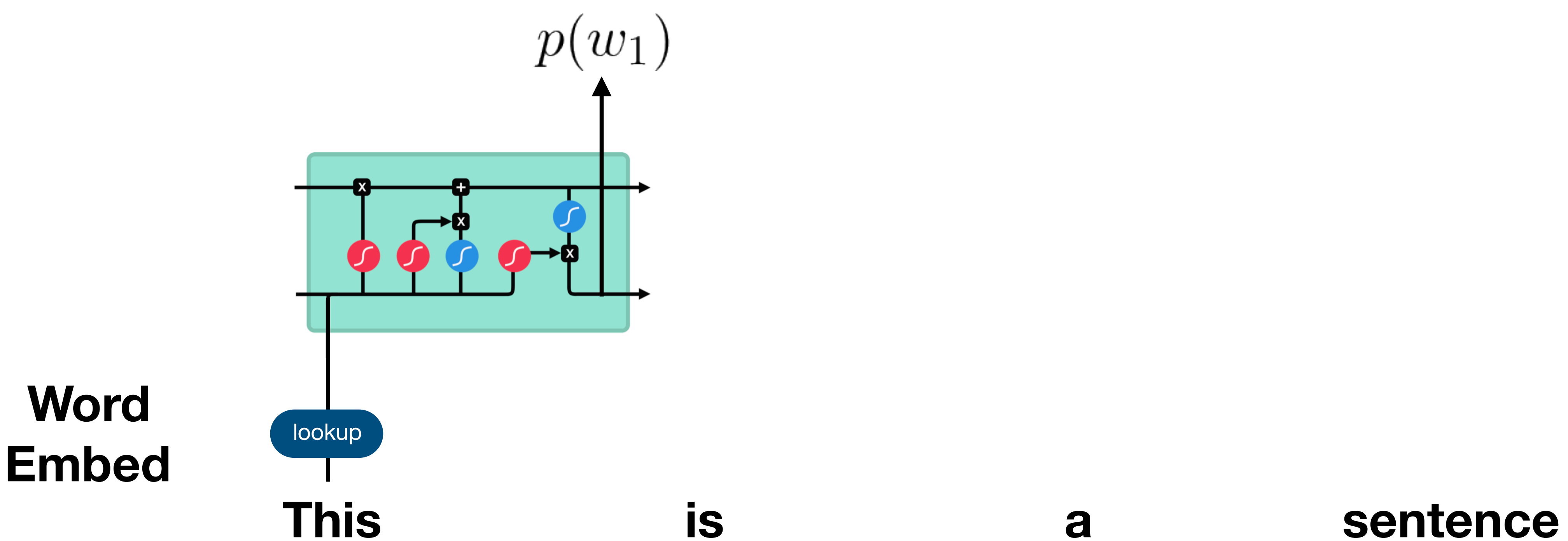


Image from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM

- Sequential in time, slow to train
- Harder to learn long-term dependencies

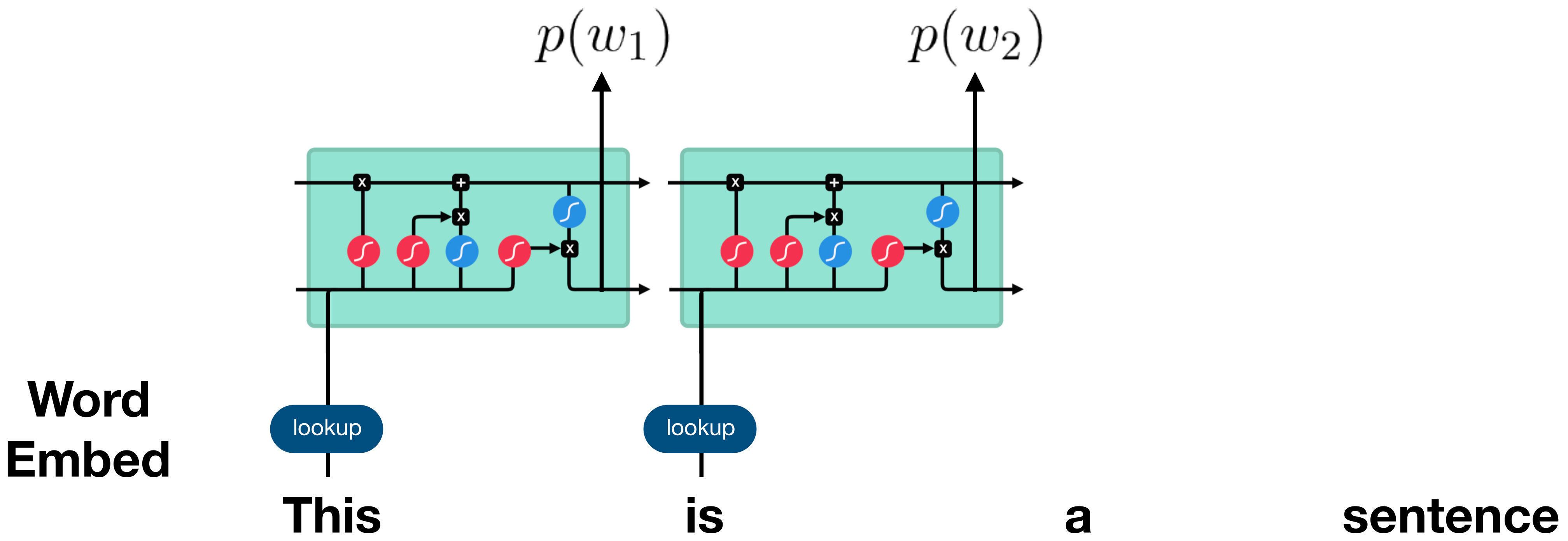


Image from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM

- Sequential in time, slow to train
- Harder to learn long-term dependencies

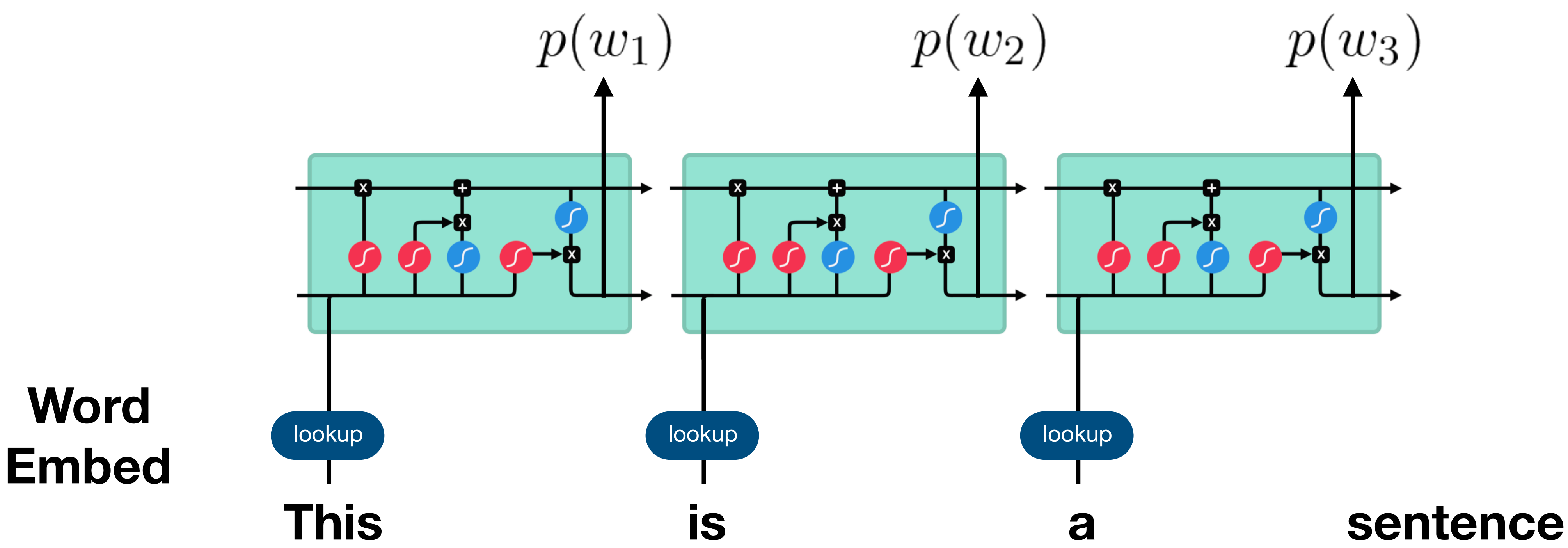


Image from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM

- Sequential in time, slow to train
- Harder to learn long-term dependencies

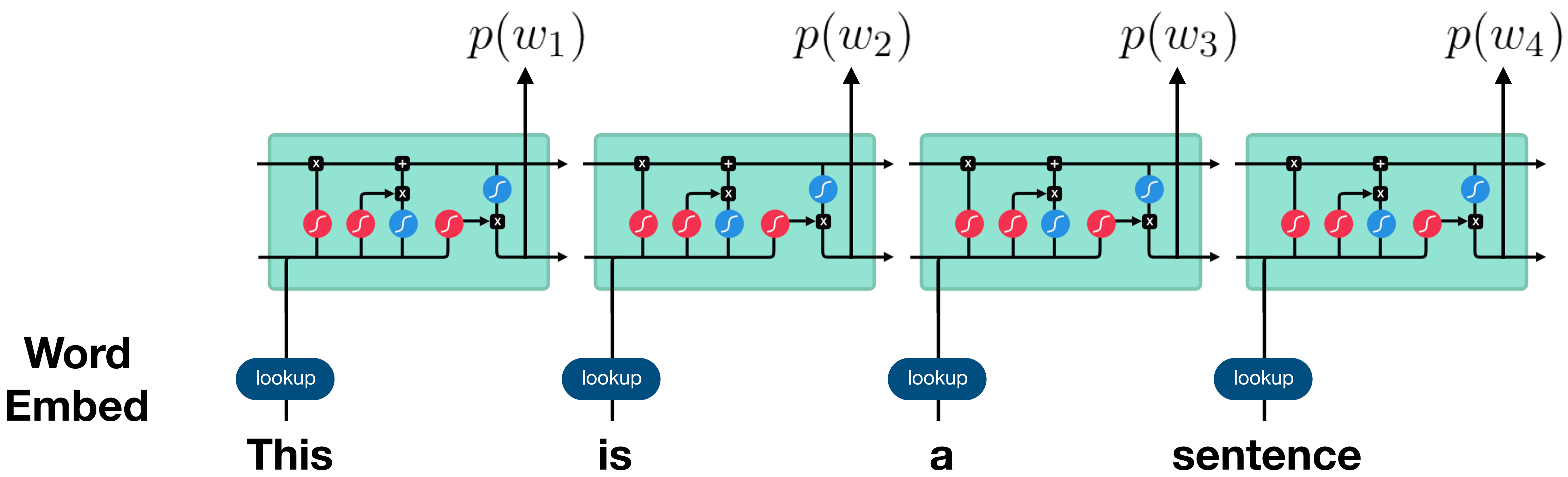


Image from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Transformer

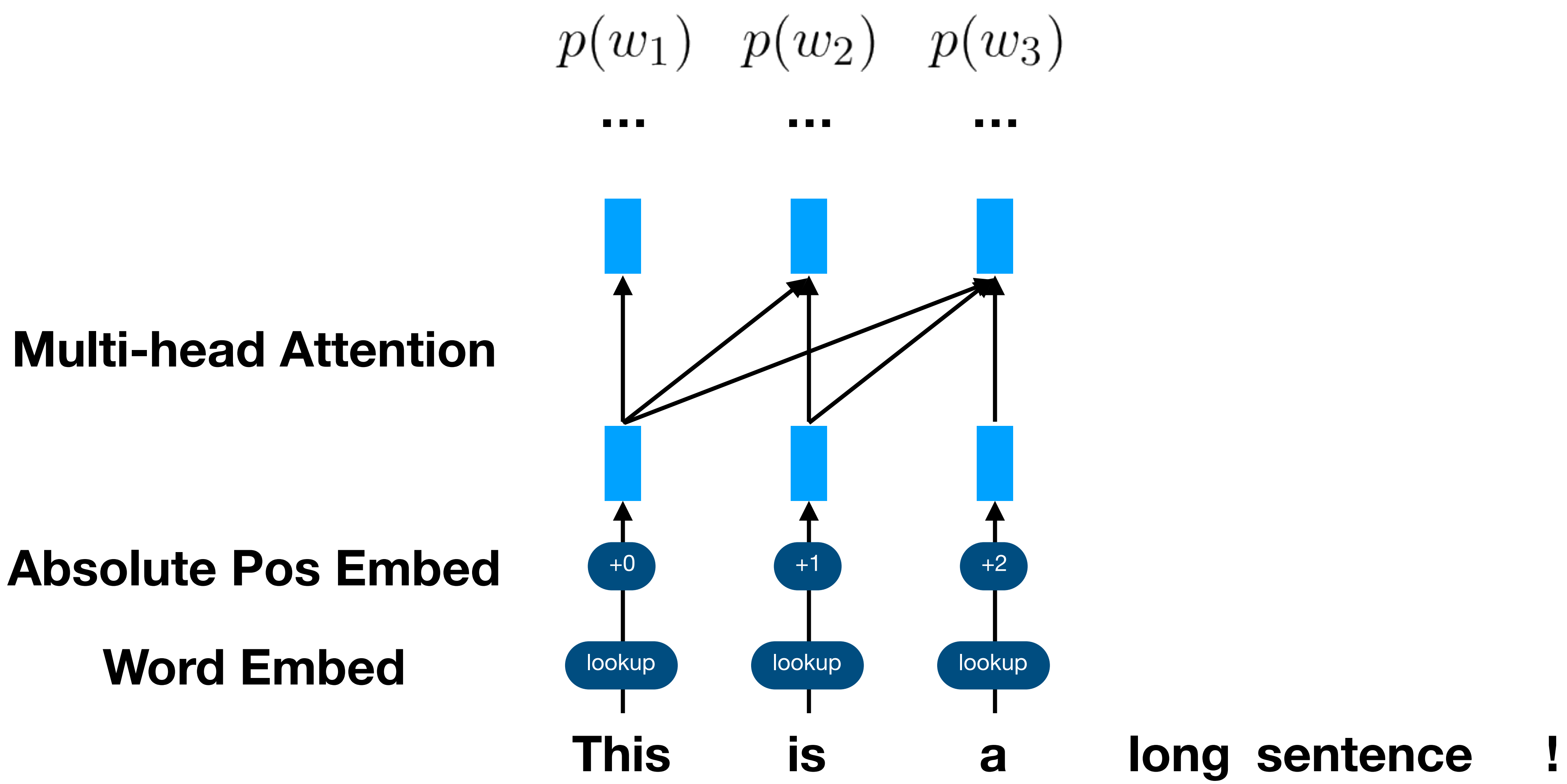
- Transformer attention is parallel in time
- Absolute position embedding → recomputation during inference

This is a long sentence !

Vaswani et al 2017

Transformer

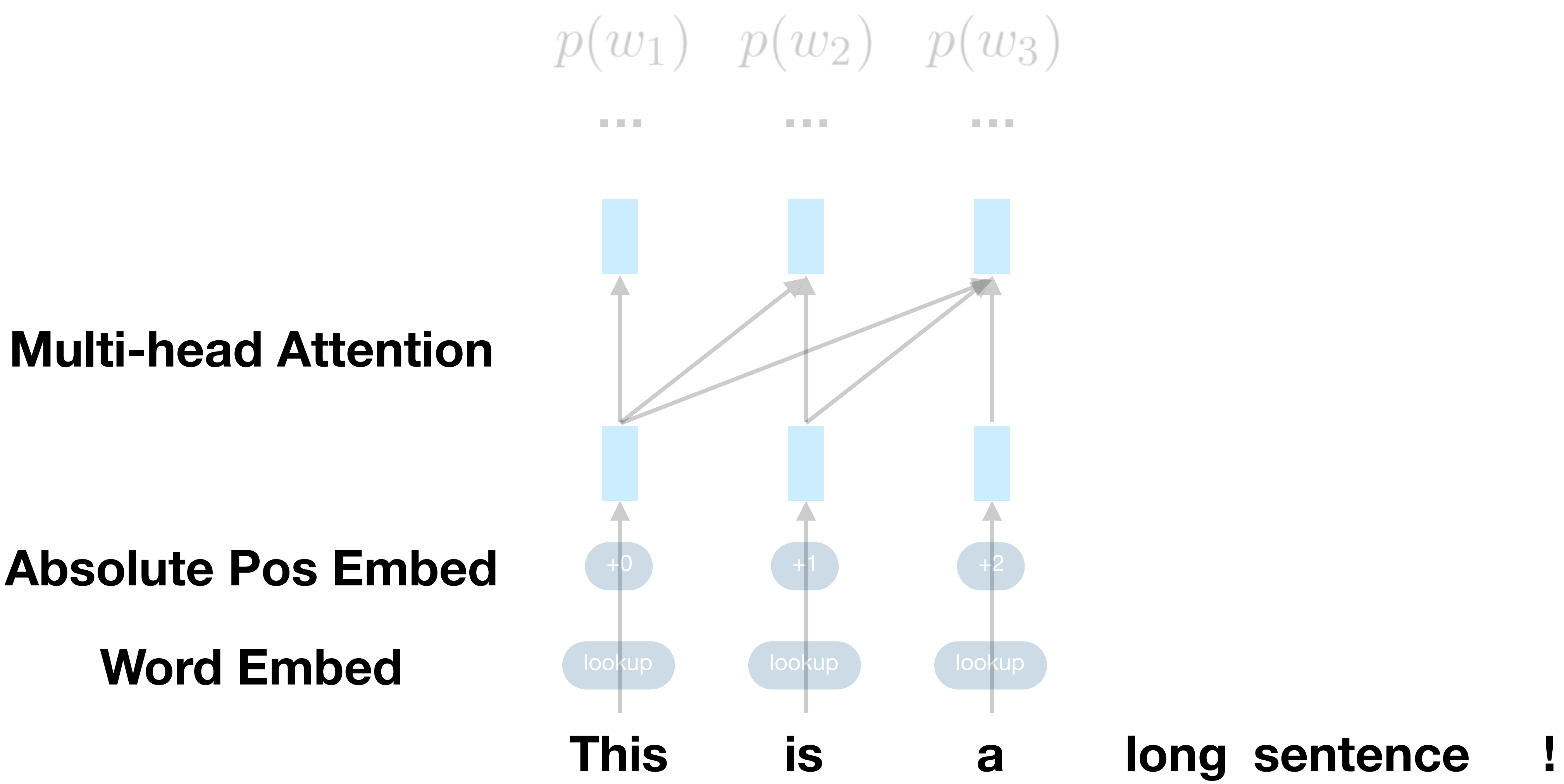
- Transformer attention is parallel in time
- Absolute position embedding → recomputation during inference



Vaswani et al 2017

Transformer

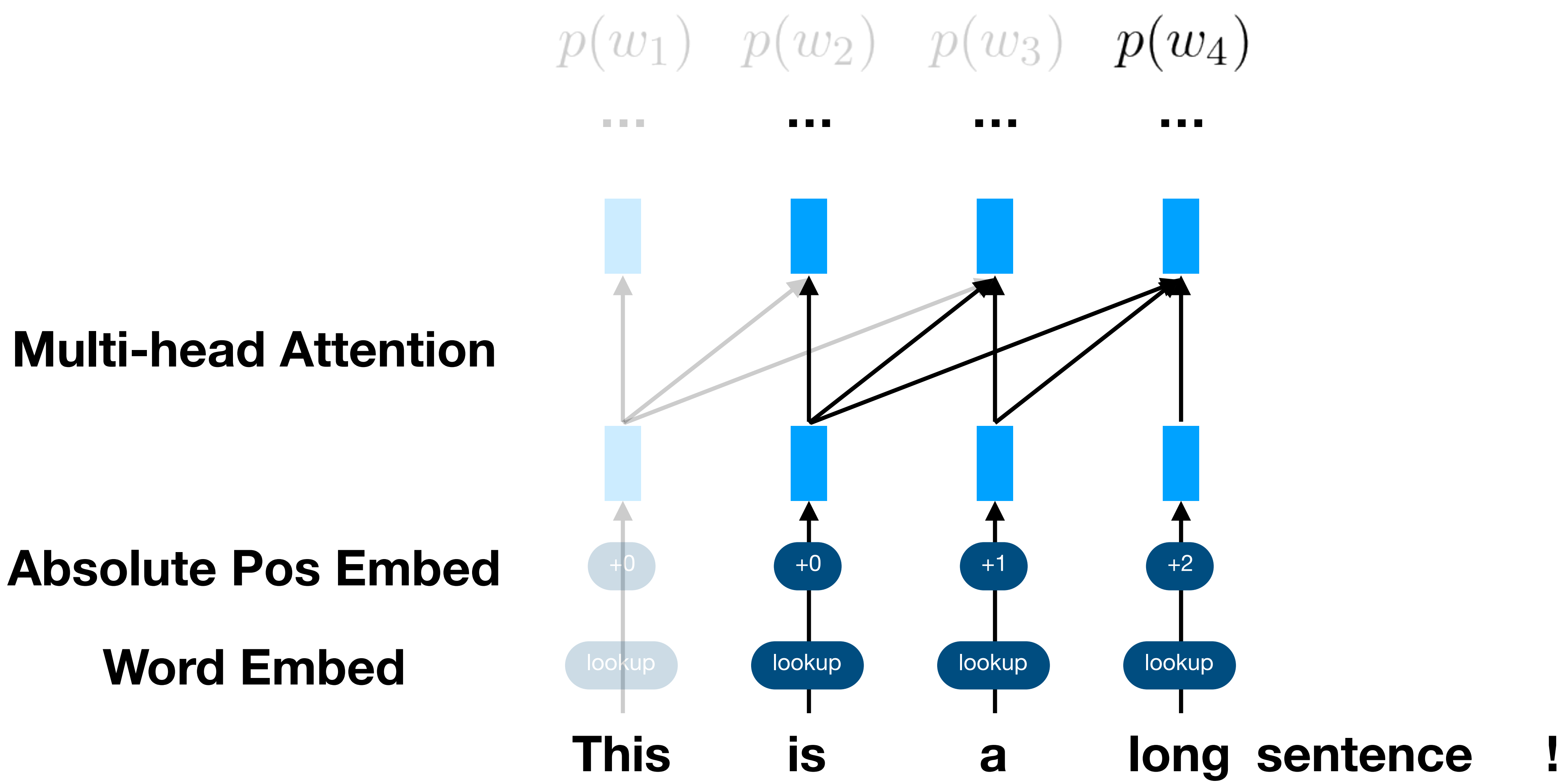
- Transformer attention is parallel in time
- Absolute position embedding → recomputation during inference



Vaswani et al 2017

Transformer

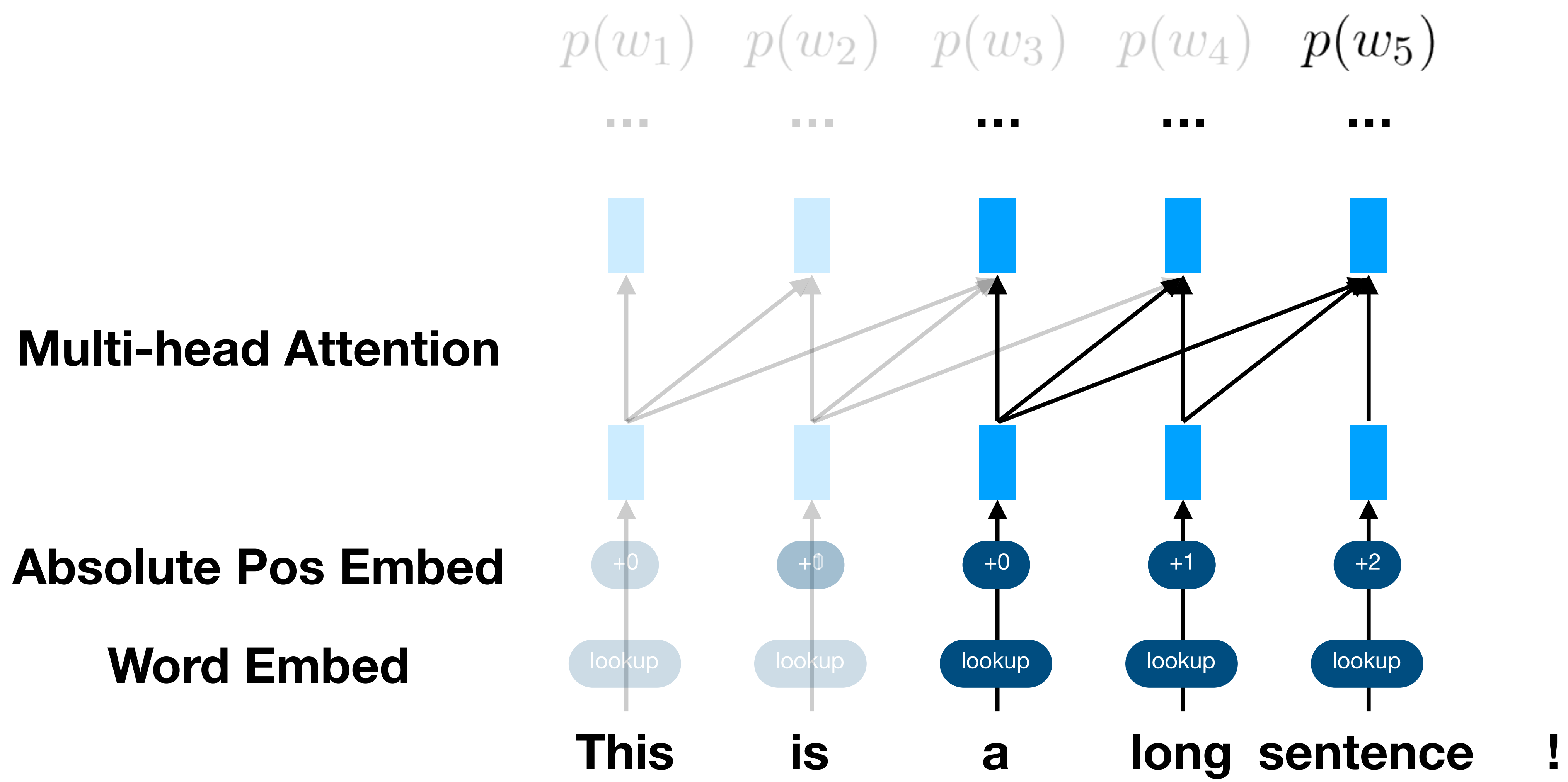
- Transformer attention is parallel in time
- Absolute position embedding → recomputation during inference



Vaswani et al 2017

Transformer

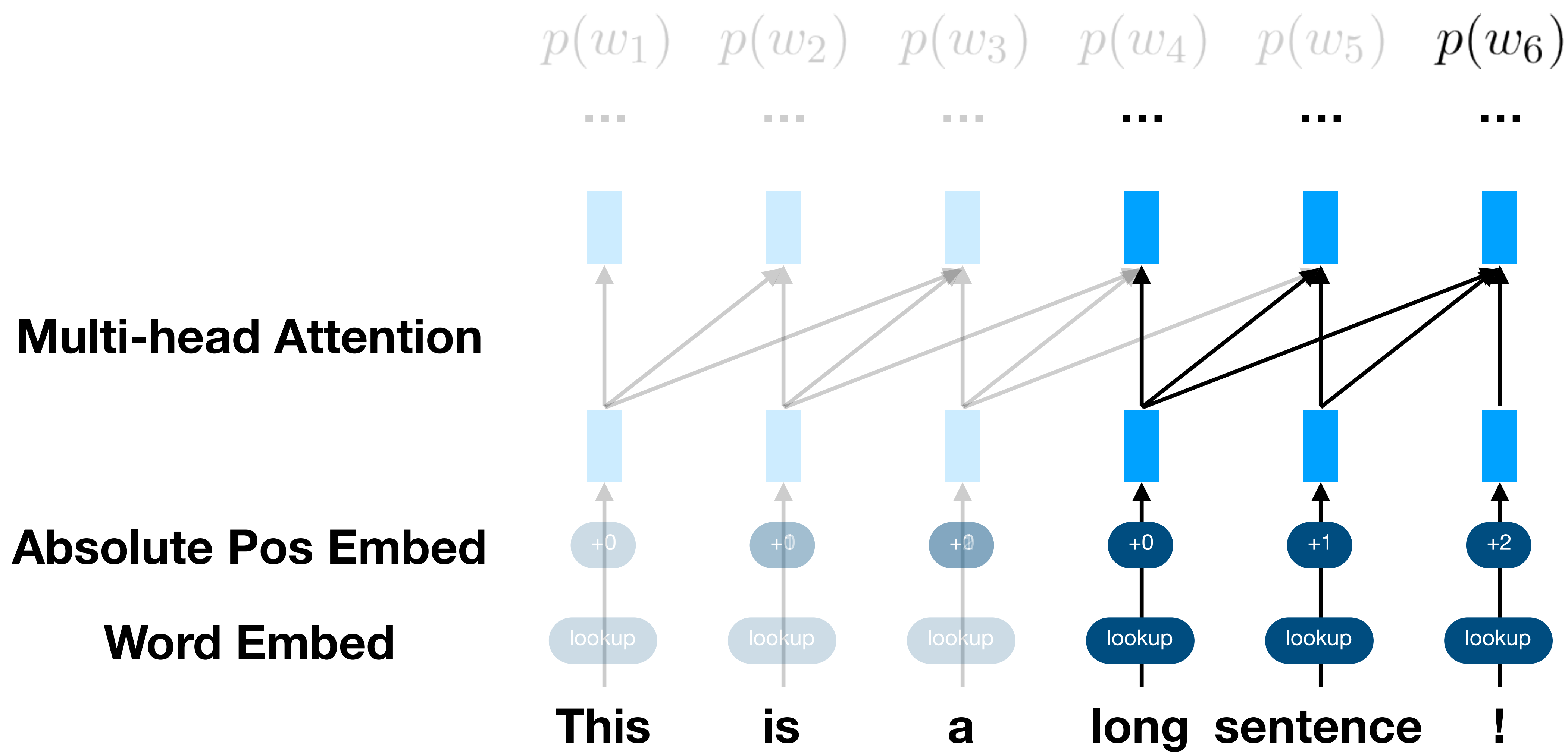
- Transformer attention is parallel in time
- Absolute position embedding → recomputation during inference



Vaswani et al 2017

Transformer

- Transformer attention is parallel in time
- Absolute position embedding → recomputation during inference



Vaswani et al 2017

Transformer-XL

- Relative position embedding → efficient inference

This is a long sentence !

Dai et al 2019

Transformer-XL

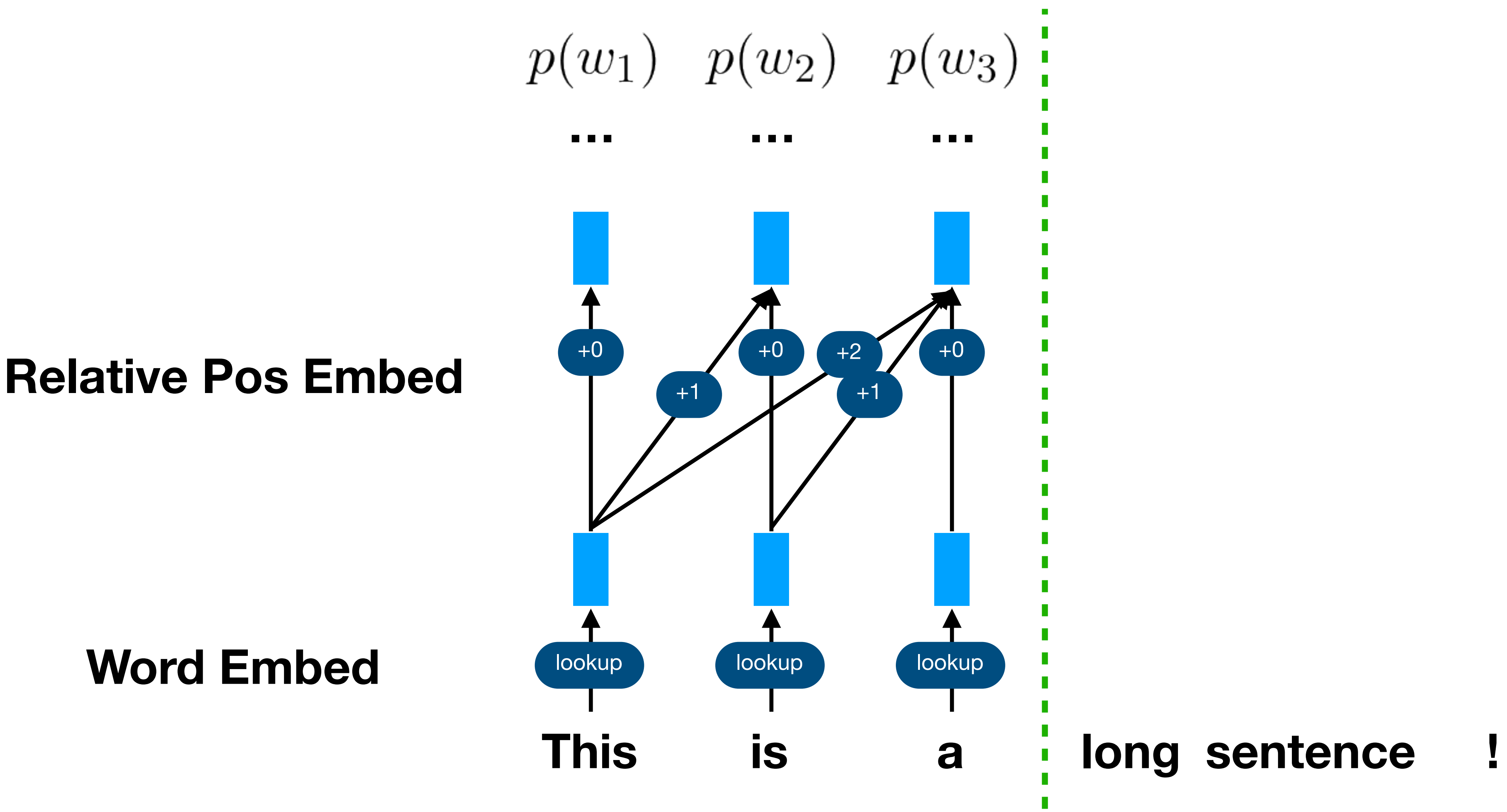
- Relative position embedding → efficient inference

This is a long sentence !

Dai et al 2019

Transformer-XL

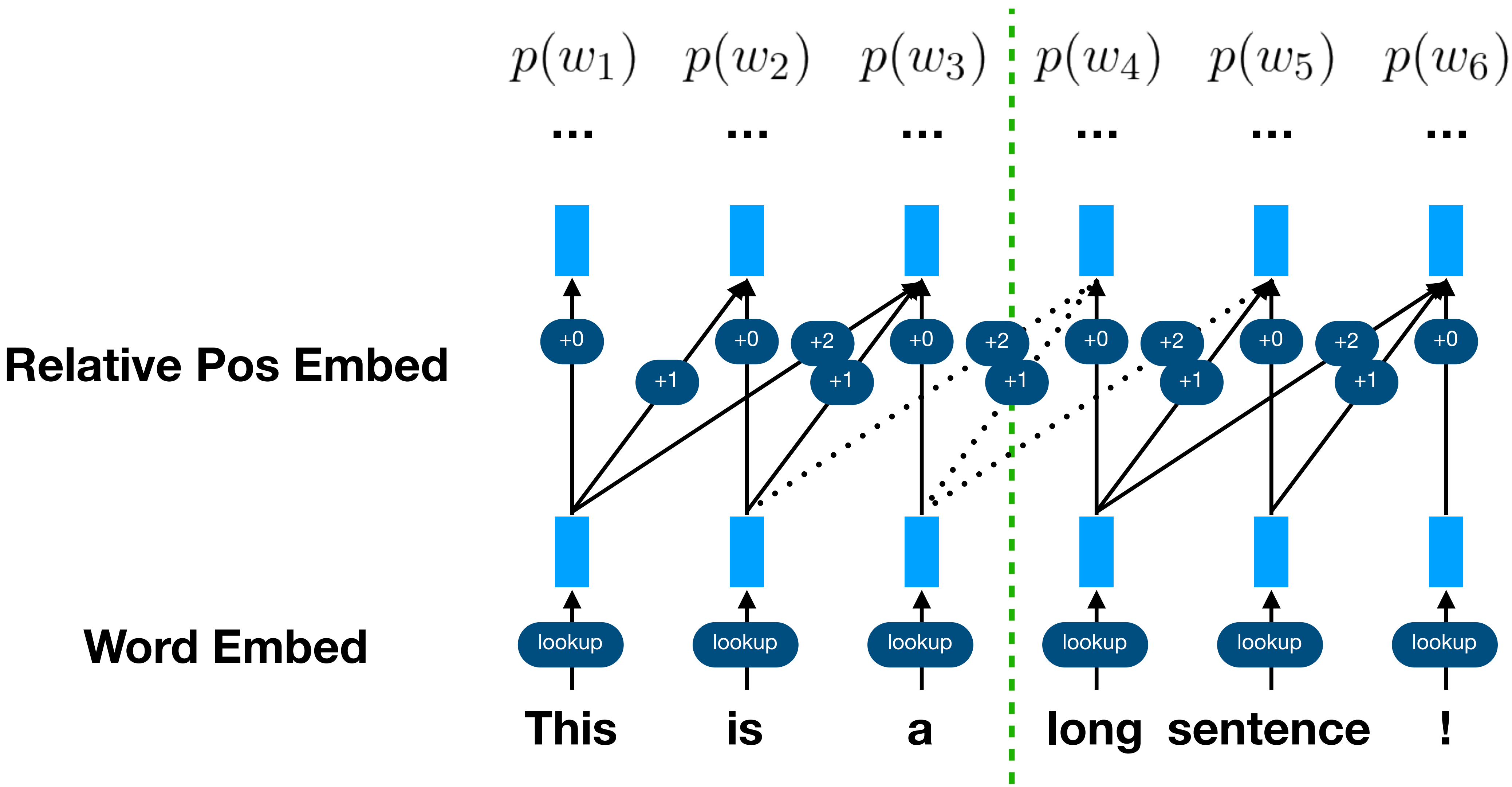
- Relative position embedding → efficient inference



Dai et al 2019

Transformer-XL

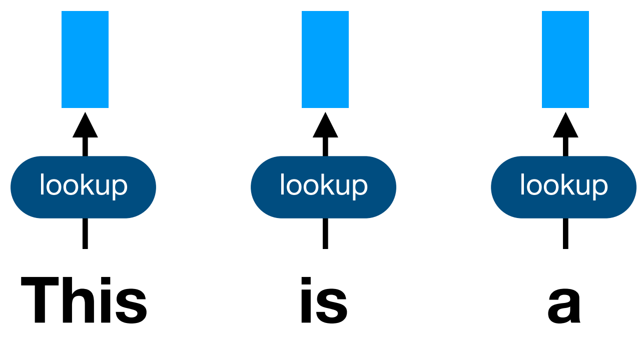
- Relative position embedding → efficient inference



Dai et al 2019

Weight-tied Nets

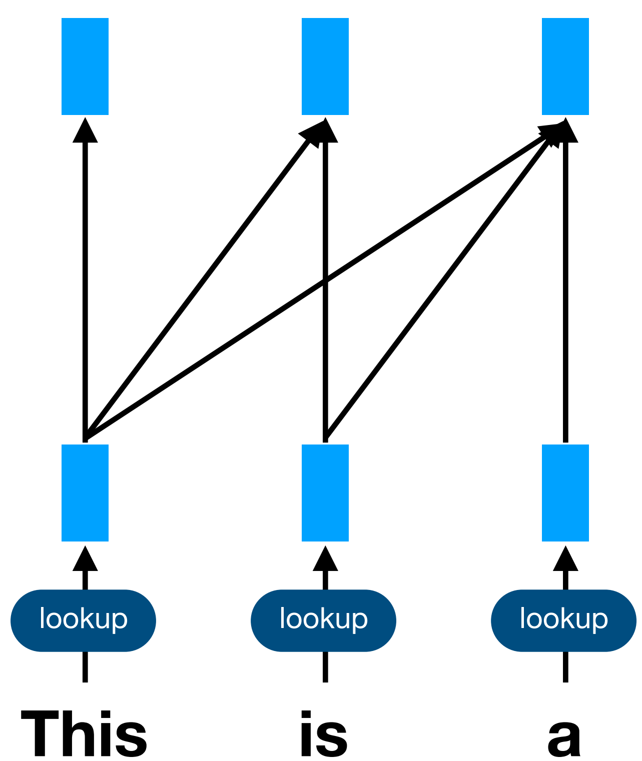
- E.g. Weight-tied transformer, Deep Equilibrium (DEQ) transformer
- Only one layer of weight
  - **But**: wider, more parameters and computation per layer



Bai et al 2019

Weight-tied Nets

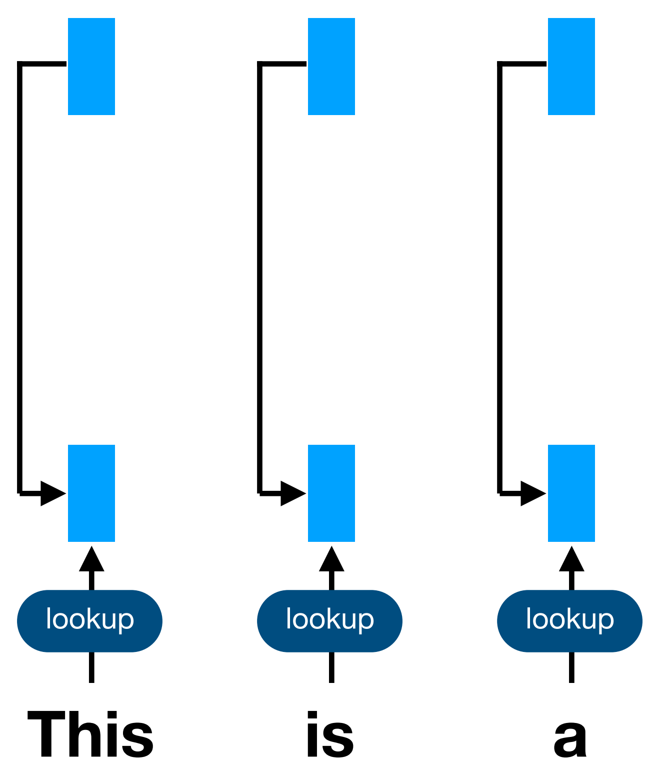
- E.g. Weight-tied transformer, Deep Equilibrium (DEQ) transformer
- Only one layer of weight
  - **But**: wider, more parameters and computation per layer



Bai et al 2019

Weight-tied Nets

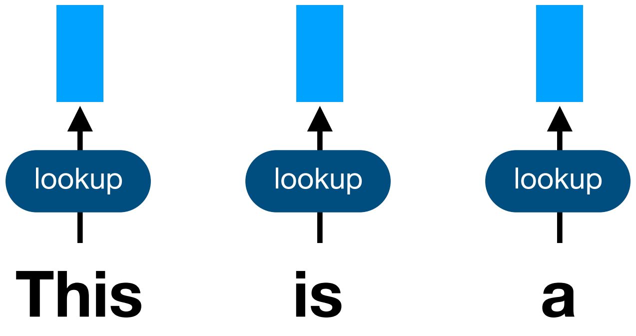
- E.g. Weight-tied transformer, Deep Equilibrium (DEQ) transformer
- Only one layer of weight
  - **But**: wider, more parameters and computation per layer



Bai et al 2019

Weight-tied Nets

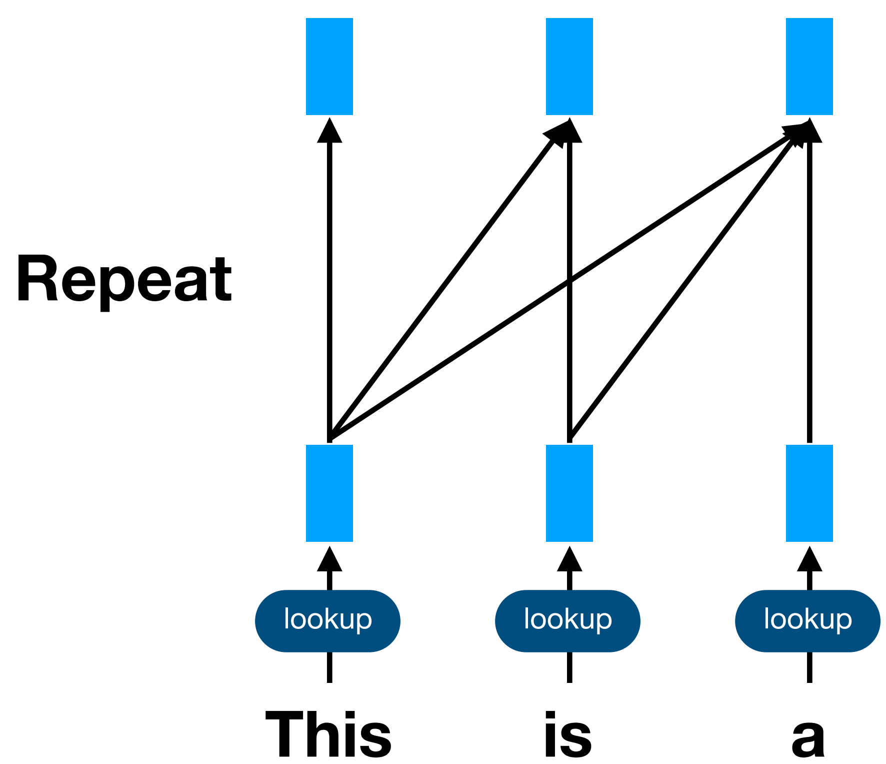
- E.g. Weight-tied transformer, Deep Equilibrium (DEQ) transformer
- Only one layer of weight
  - **But**: wider, more parameters and computation per layer



Bai et al 2019

Weight-tied Nets

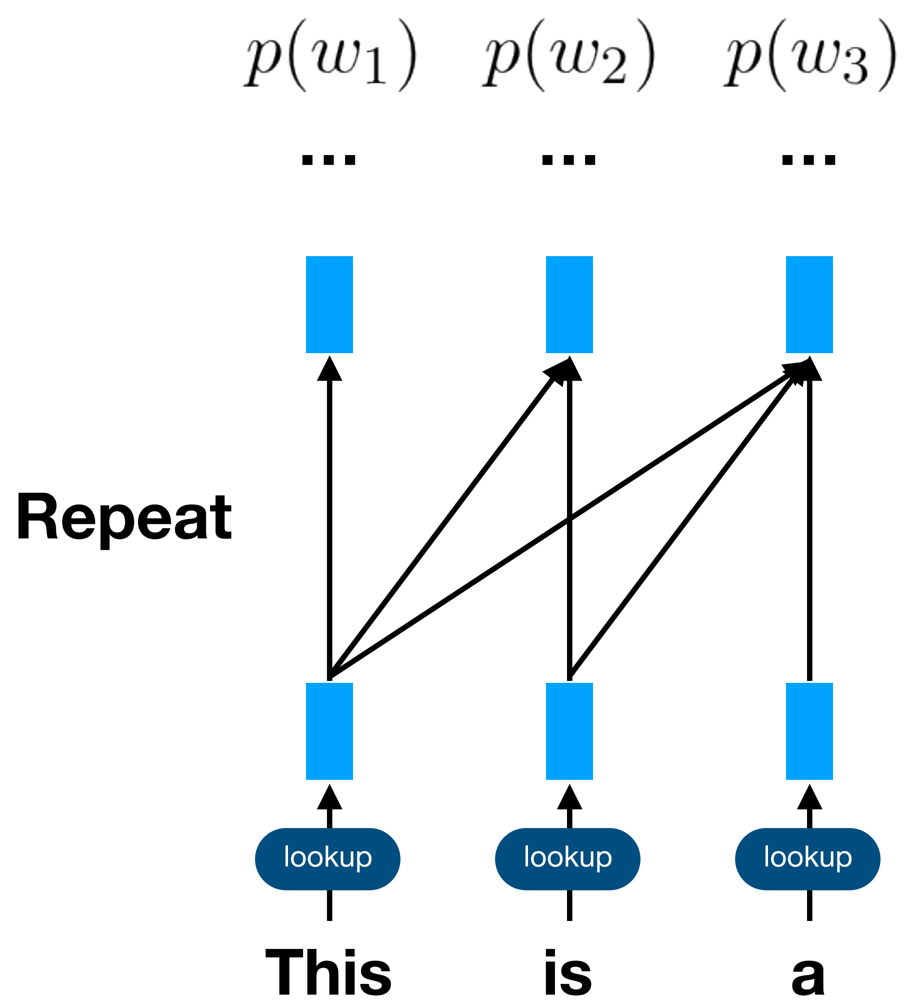
- E.g. Weight-tied transformer, Deep Equilibrium (DEQ) transformer
- Only one layer of weight
  - **But**: wider, more parameters and computation per layer



Bai et al 2019

Weight-tied Nets

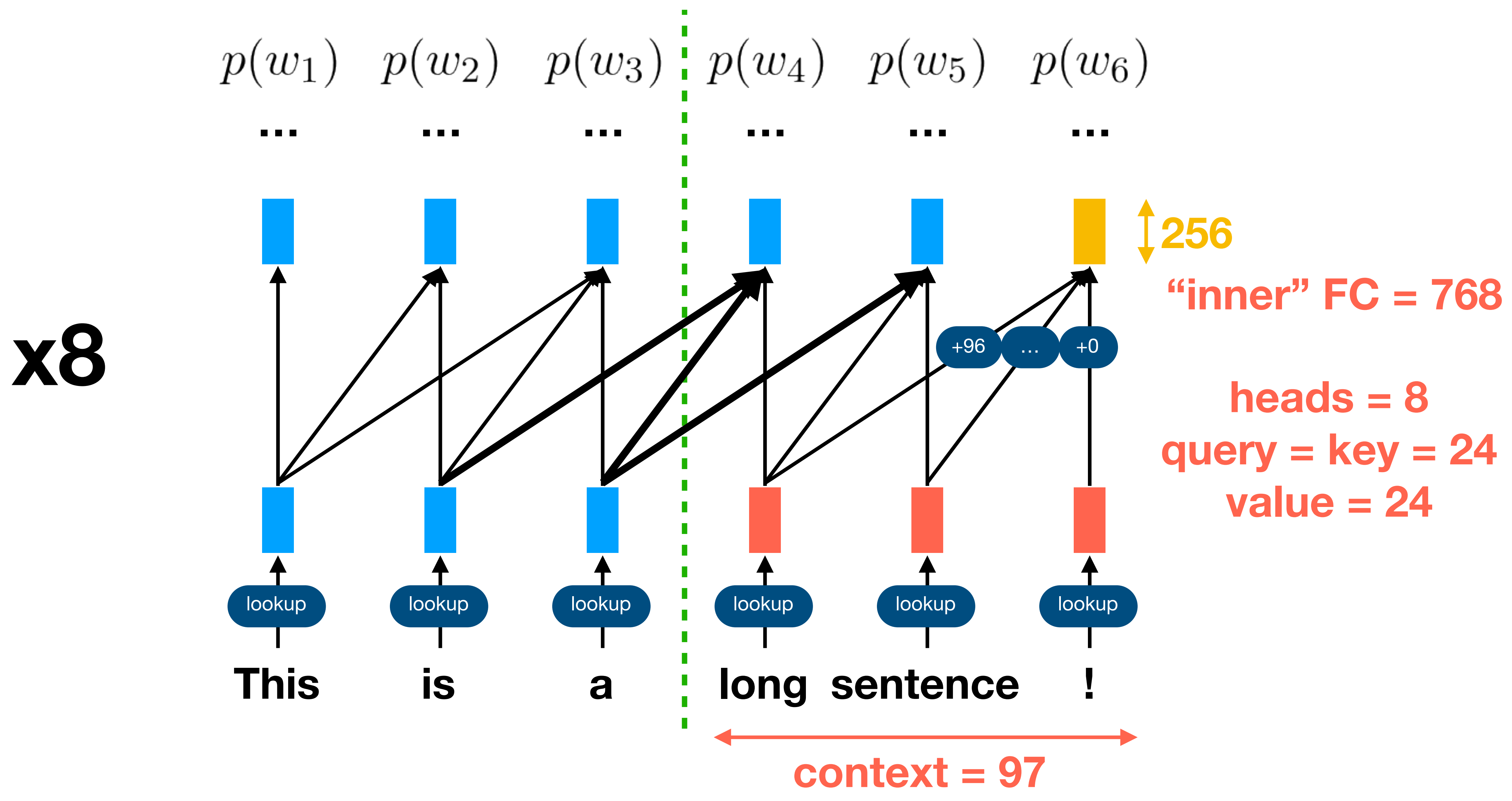
- E.g. Weight-tied transformer, Deep Equilibrium (DEQ) transformer
- Only one layer of weight
  - **But**: wider, more parameters and computation per layer



Bai et al 2019

Our Base Model

- Fully differentiable transformer-XL

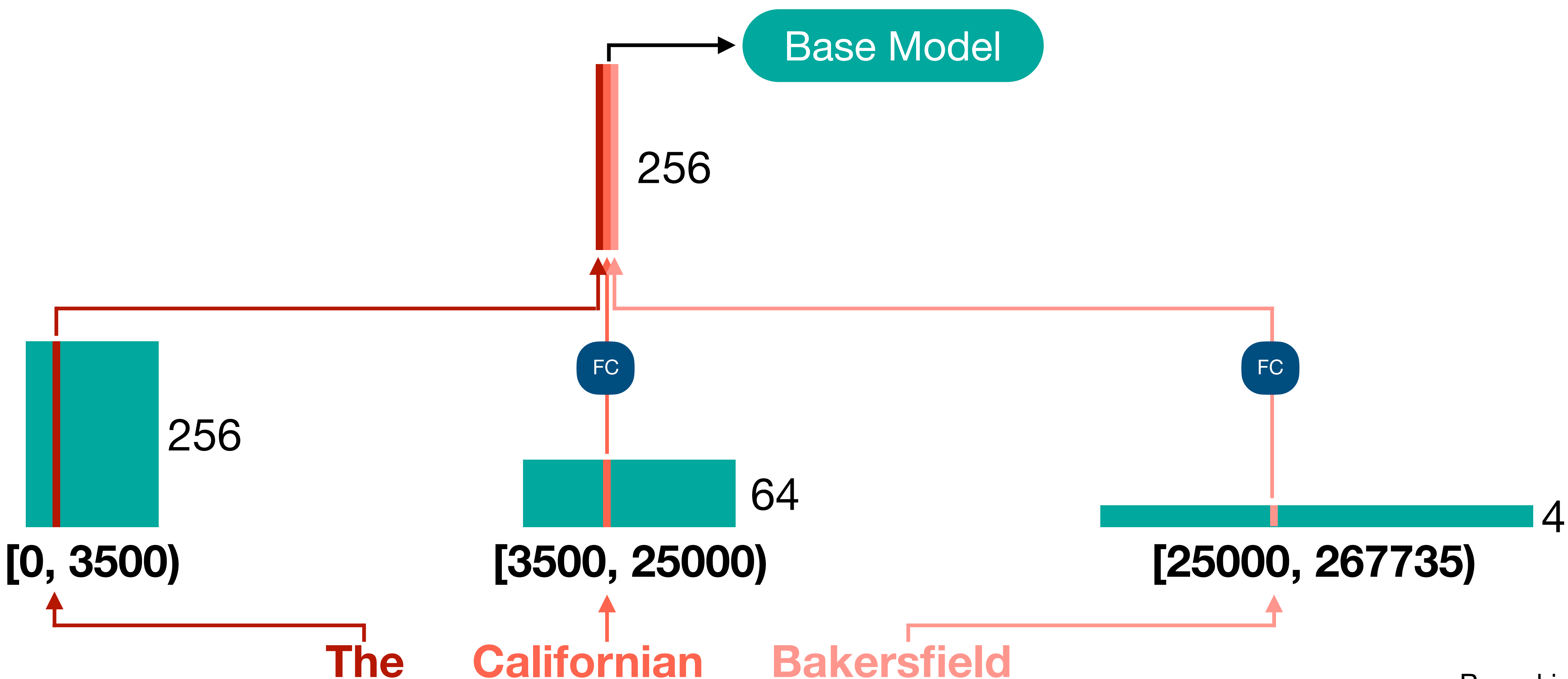


Observations

- Large vocab size: 267735
- Naive word embedding: 267735 x 256 = 69M params
  - 138M mul/add's at output embedding
- Rare words need less representation!

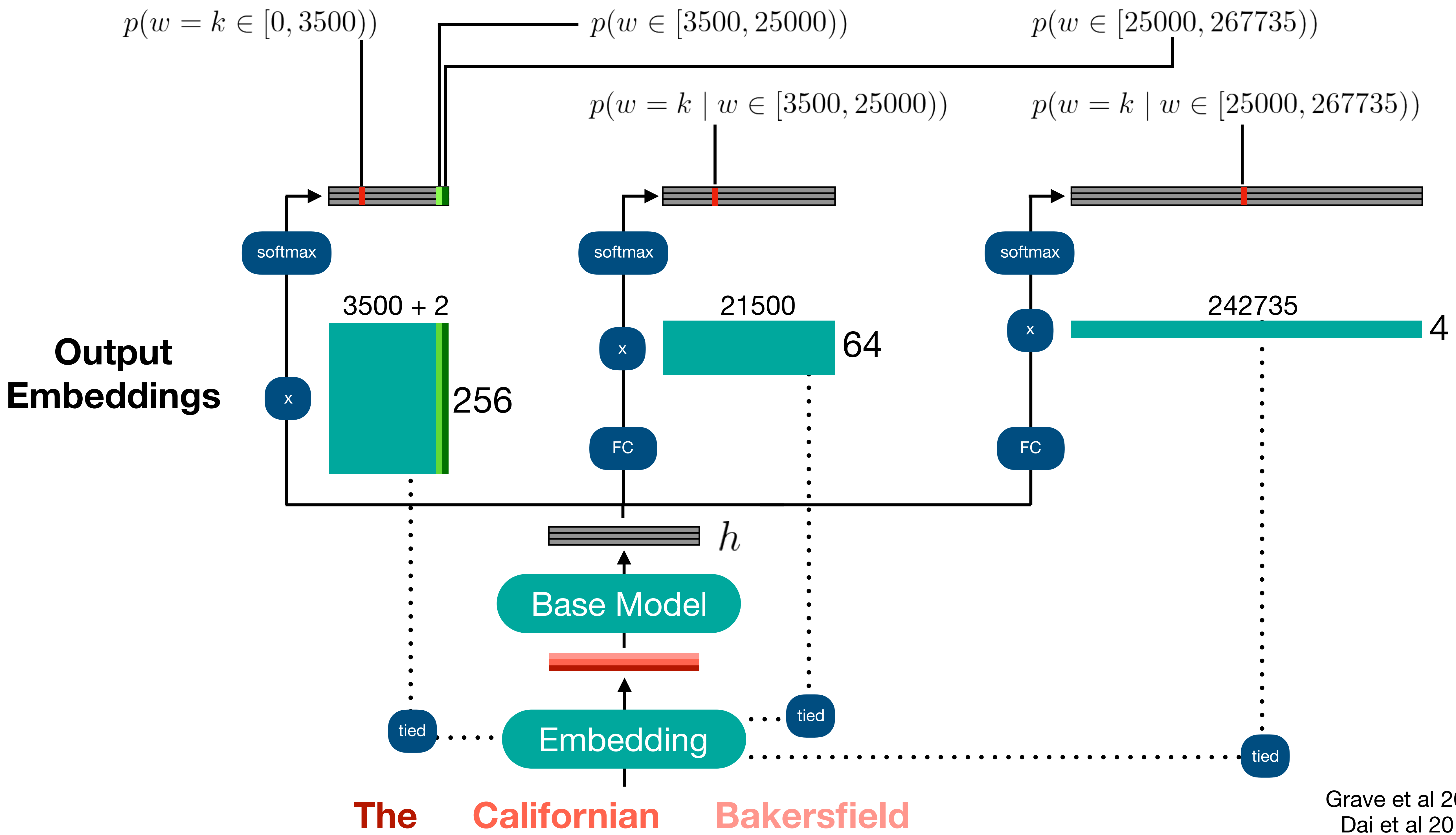
Adaptive Word Embedding

- Sort vocabulary and divide into 3 bins by frequency
- Low-rank decomposition



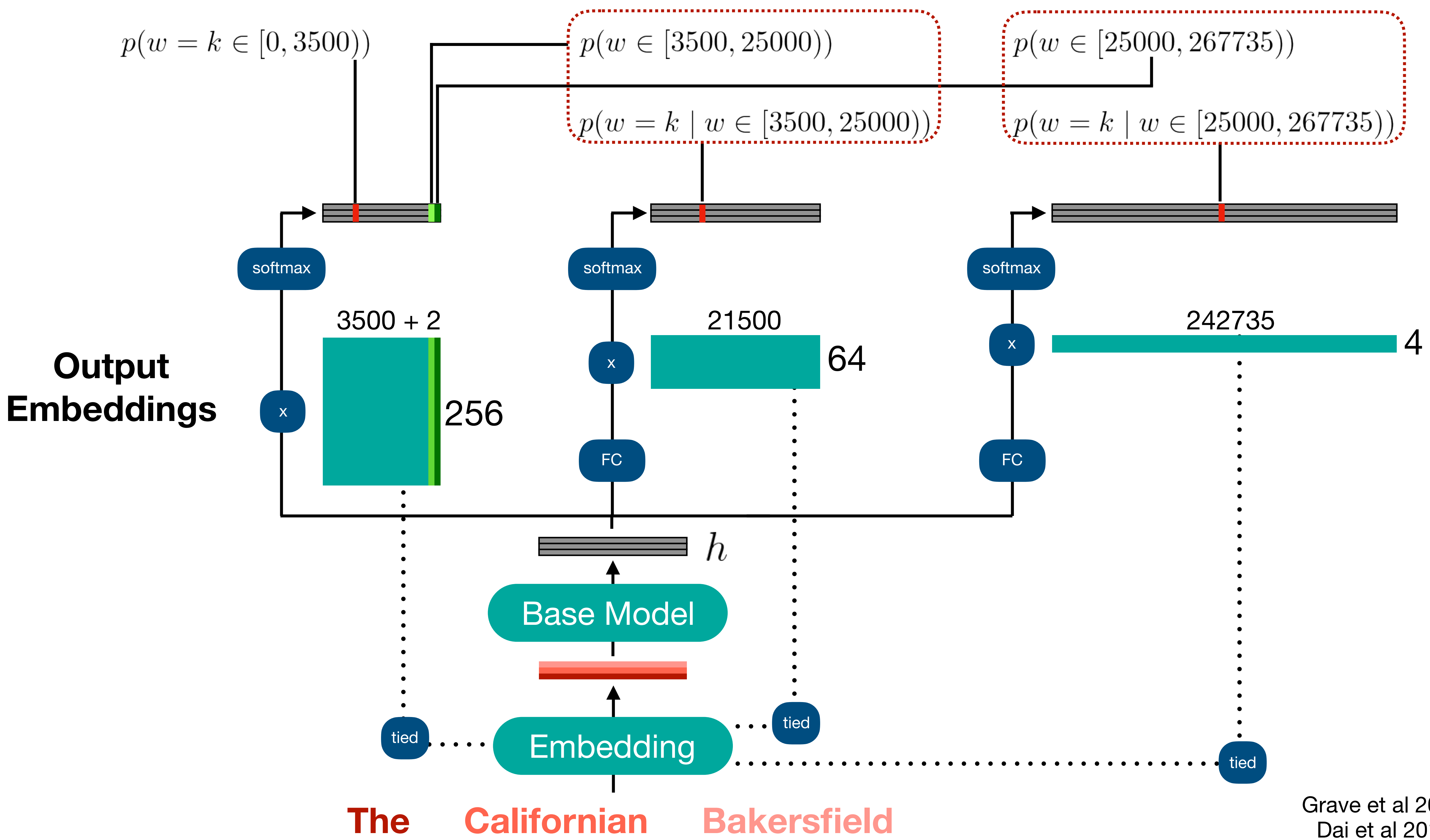
Baevski et al 2018
Dai et al 2019

Adaptive Softmax



Grave et al 2016
Dai et al 2019

Adaptive Softmax



Grave et al 2016
Dai et al 2019

Performance

- Learning rate = 1e-3
- Cosine learning rate decay
- 1 RTX 2080 Ti → ~1 day
- Compute ~ 2 * Params

Layers	Embed Param (M)	Param (M)	Val PPL	Q, K, V	Inner
5	3.3	7.8	39.4	48	1024
6	3.3	8.3	37.9	32	1024
8	**3.3**	**8.3**	**37.2**	**24**	**768**
LSTM Baseline*	138	159	29.0		

* Baseline is Rae et al 2018

Observations

- Rare words have a much higher probability of recurring than occurring

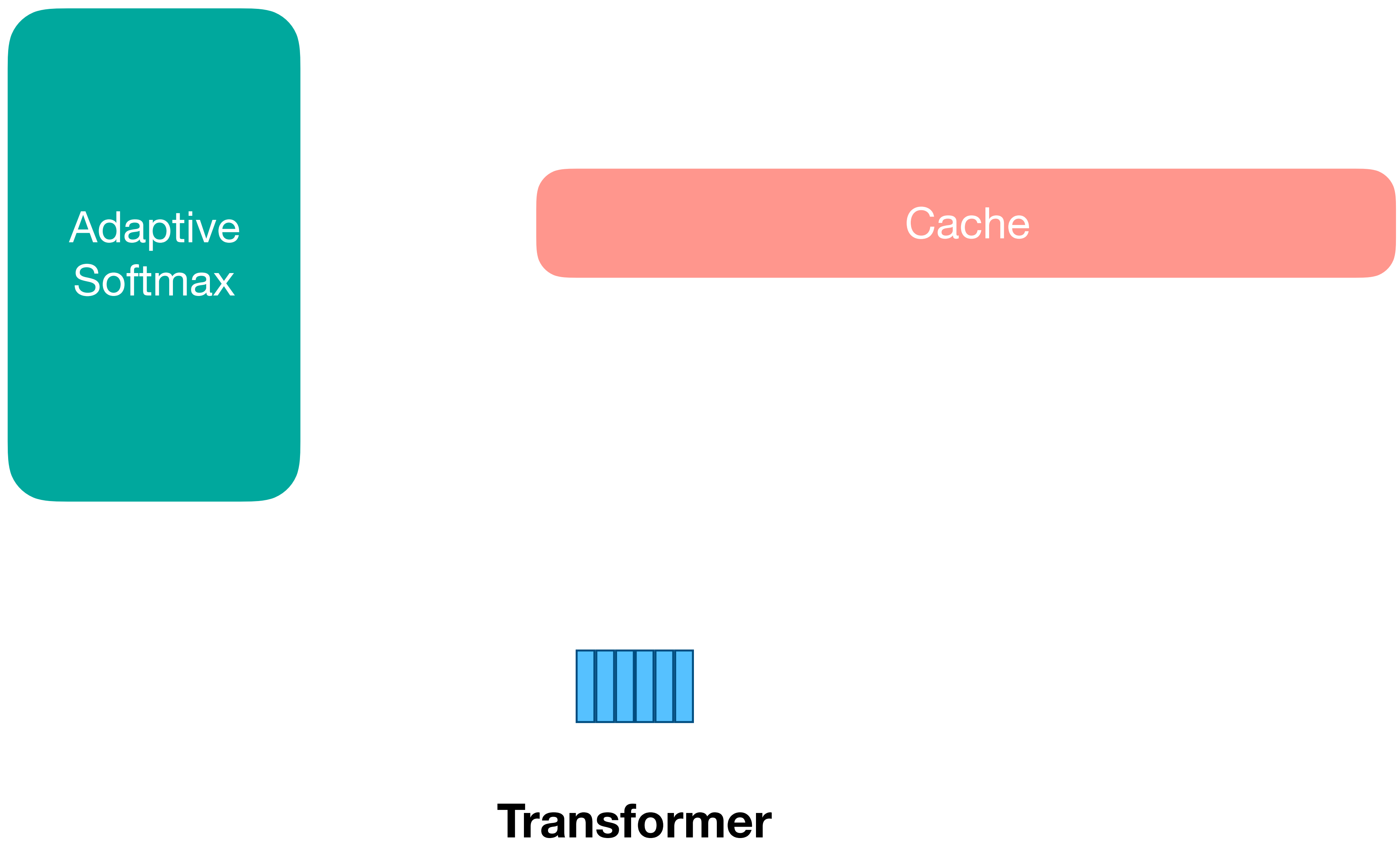
Vancouver was originally named Gastown and began as a settlement which grew around the site of a makeshift tavern on the western edges of Hastings Mill built on July 1, 1867, and owned by proprietor Gassy Jack. The original site is marked by the Gastown steam clock. Gastown then formally registered as a townsite dubbed Granville, Burrard Inlet.

- How to compensate for short attention context (97)?

Differentiable Cache

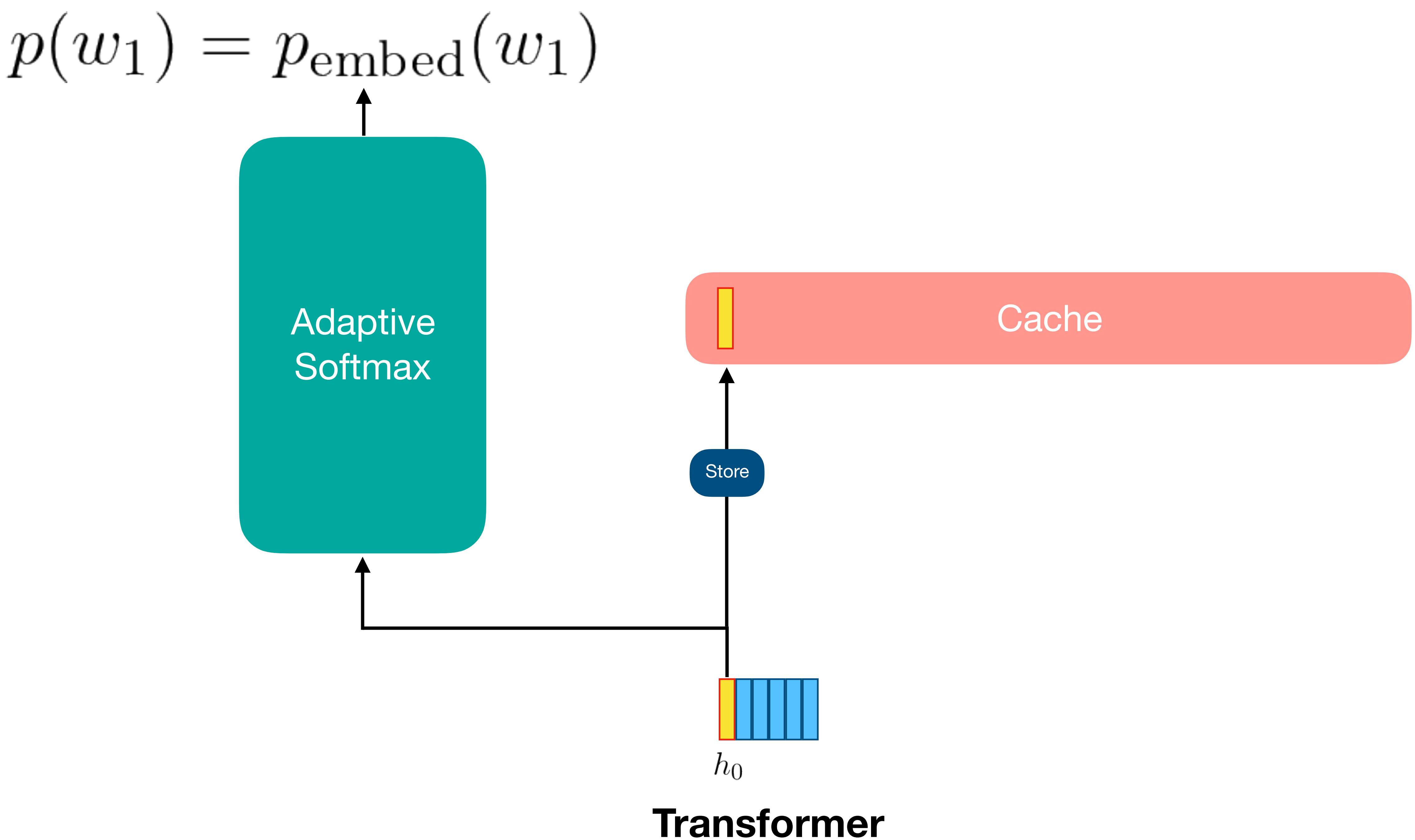
- Keep most recent activations in differentiable cache!
  - 0 parameters
  - Adds some computation

Differentiable Cache



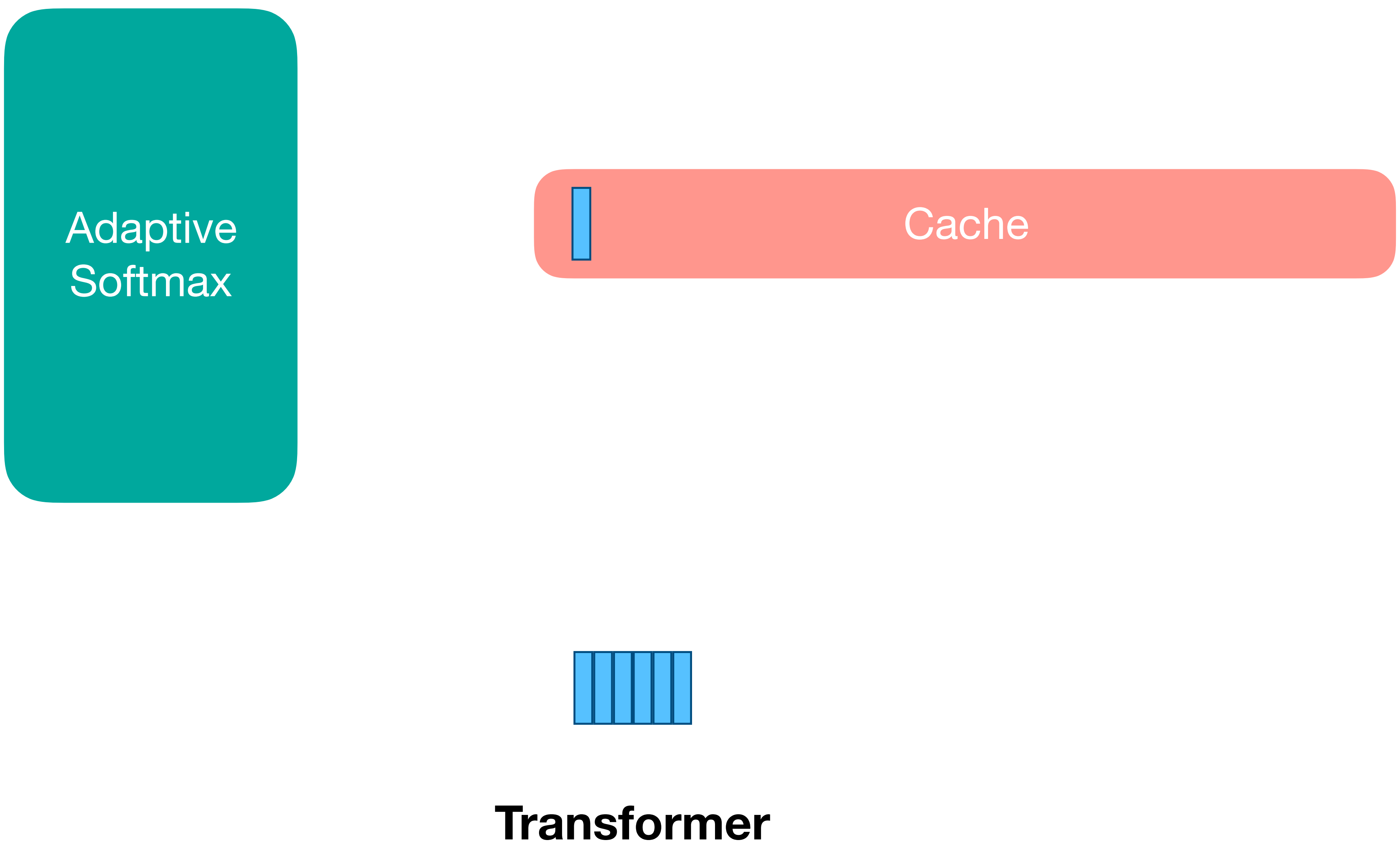
Grave et al 2016
Rae et al 2018

Differentiable Cache



Grave et al 2016
Rae et al 2018

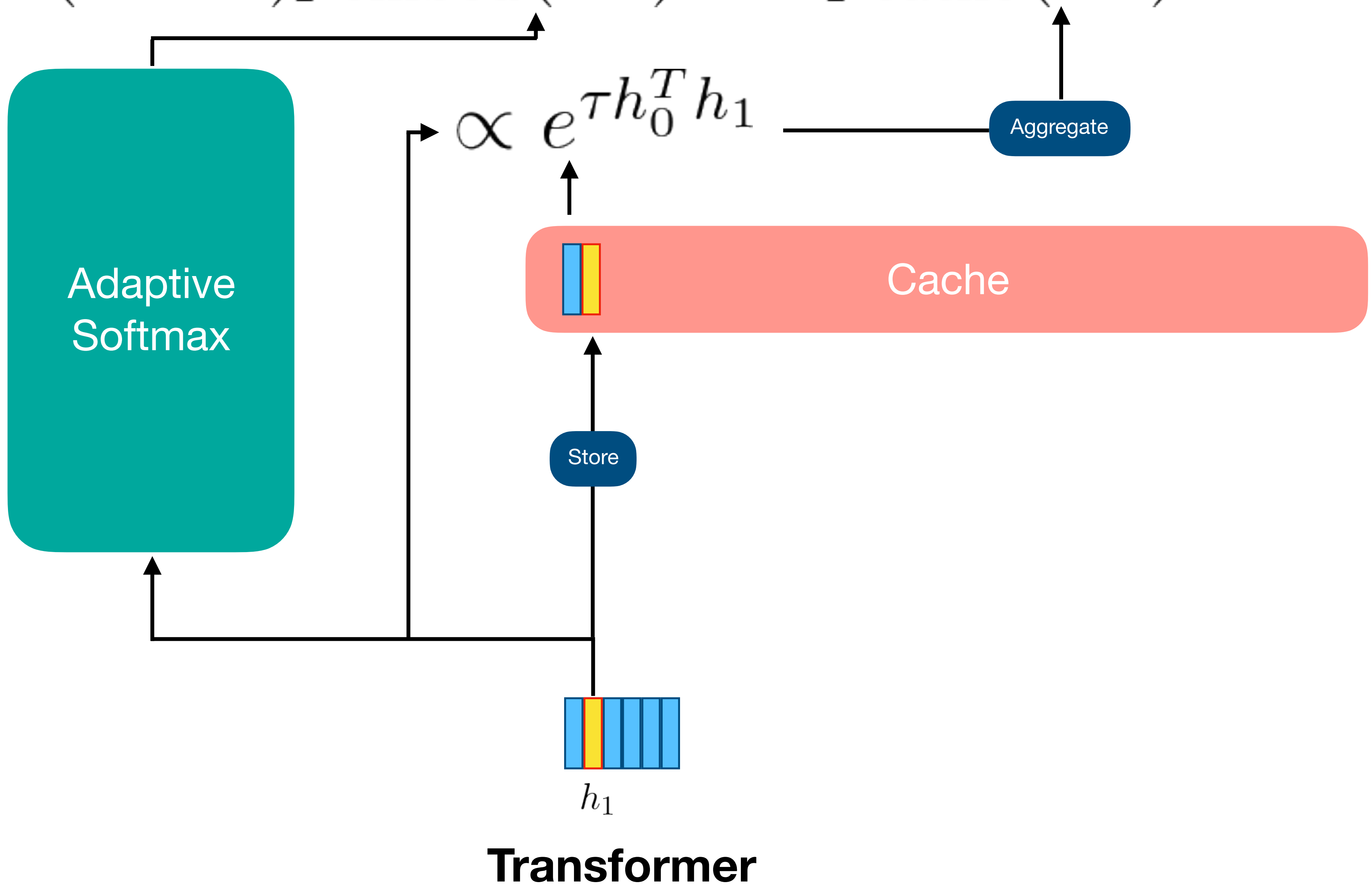
Differentiable Cache



Grave et al 2016
Rae et al 2018

Differentiable Cache

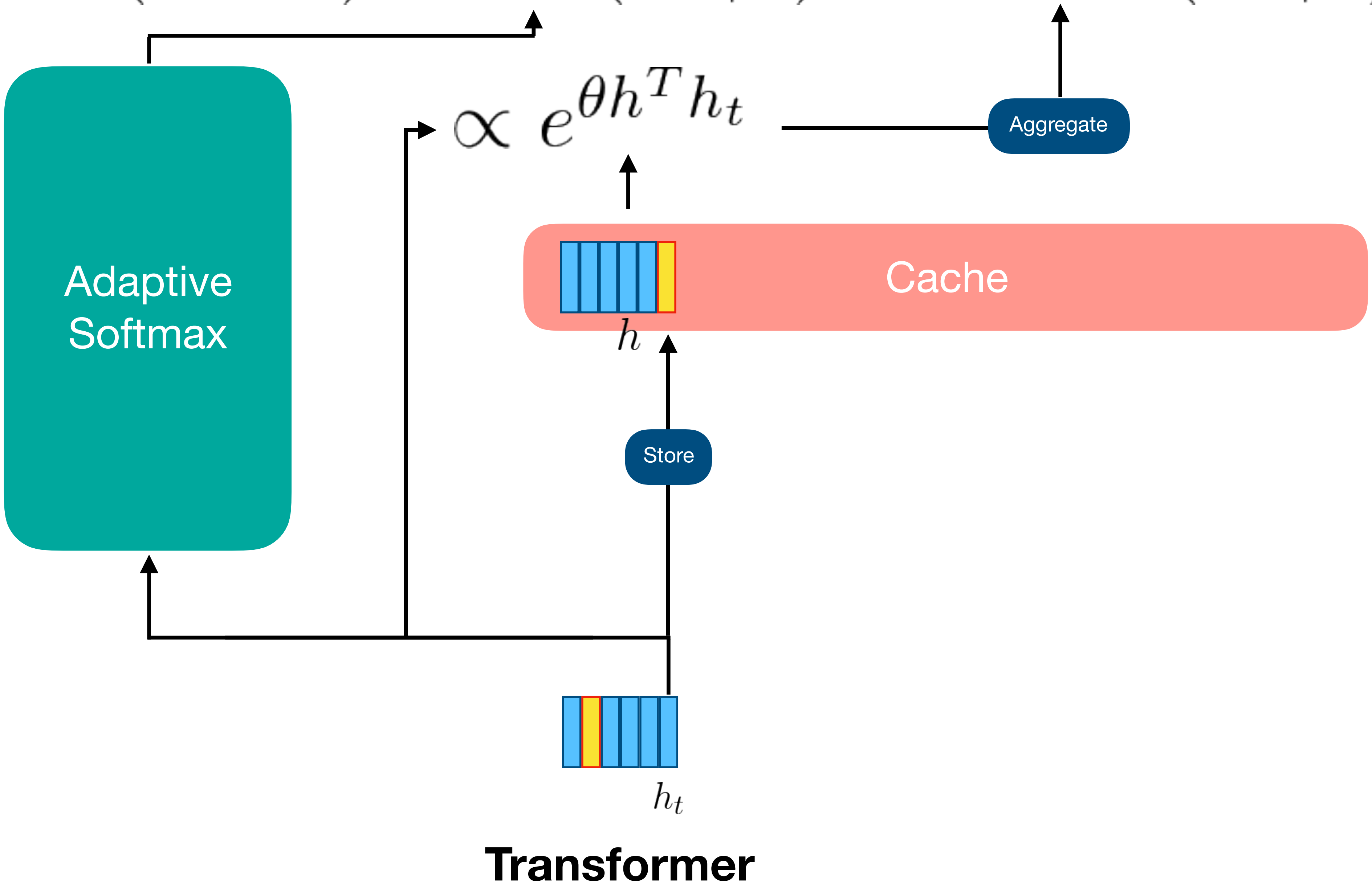
$$p(w_2) = (1 - \lambda) p_{\text{embed}}(w_2) + \lambda p_{\text{cache}}(w_2)$$



Grave et al 2016
Rae et al 2018

Differentiable Cache

$$p(w_{t+1}) = (1-\lambda)p_{\text{embed}}(w_{t+1}) + \lambda p_{\text{cache}}(w_{t+1})$$



Grave et al 2016
Rae et al 2018

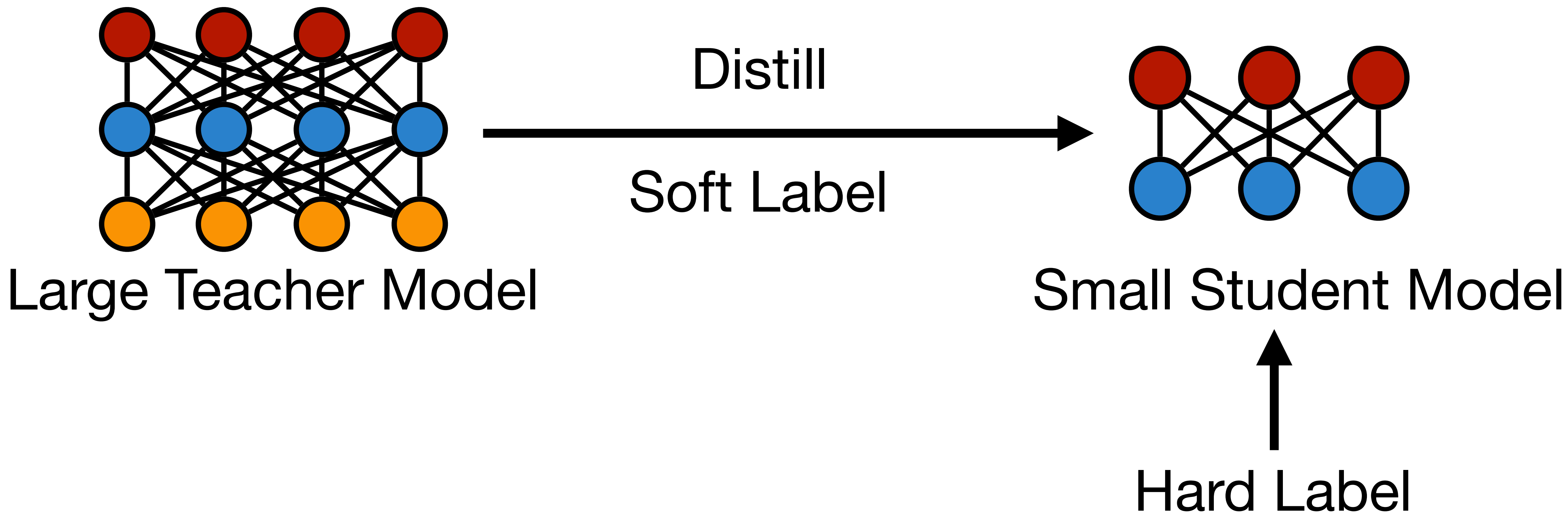
Performance

Param	Cache	Val PPL
8.3M	0	37.2
11.0M	0	35.3
15.2M	0	31.5
18.1M	0	30.1
8.3M	1000	33.1
8.3M	**2000**	**32.0**

Model Compression

- Knowledge distillation
- Pruning
- Quantization

Knowledge Distillation



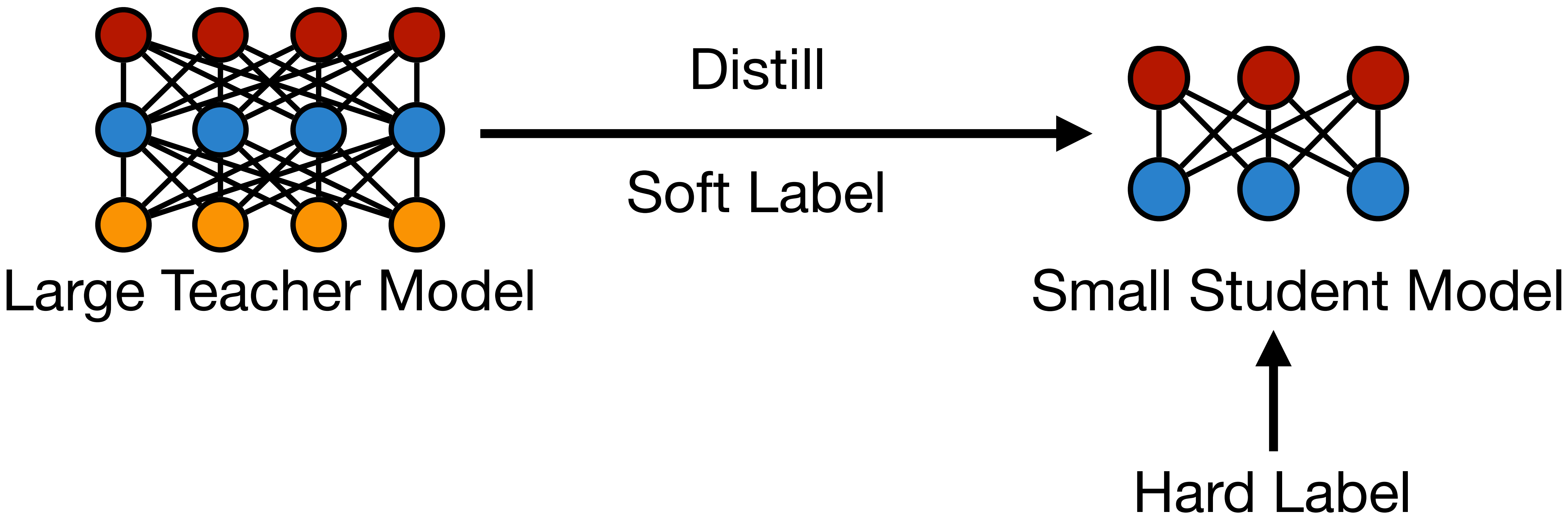
Knowledge Distillation

- Large teacher: 53M params, Test PPL 22

Knowledge Distillation

- Large teacher: 53M params, Test PPL 22
- Student: 8.3M params

Knowledge Distillation



- Large teacher: 53M params, Test PPL 22
- Student: 8.3M params
- Top 30 soft labels

Knowledge Distillation

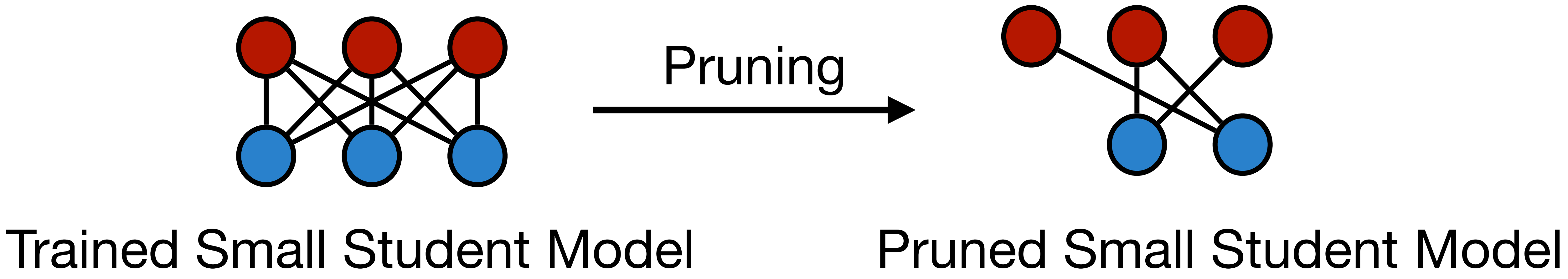
- Large teacher: 53M params, Test PPL 22
- Student: 8.3M params
- Top 30 soft labels
- Teacher annealing, first learn from teacher then learn from ground truth

Performance

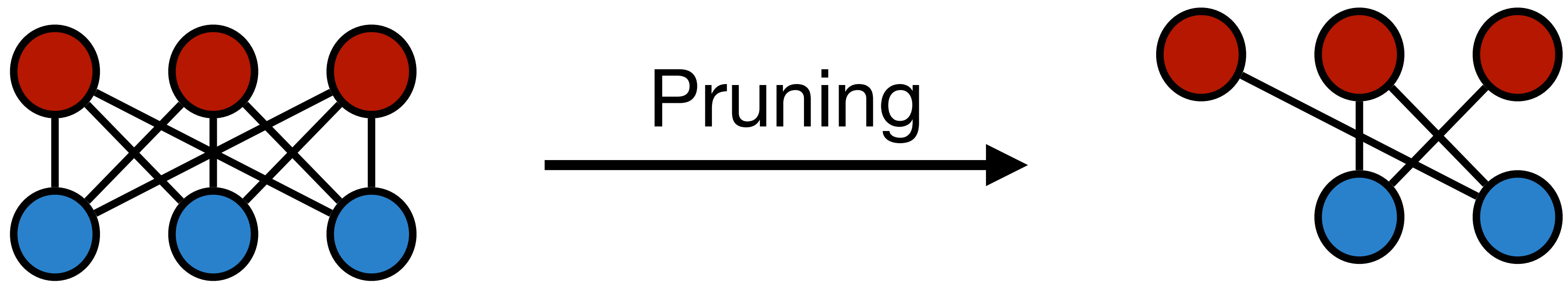
Model	Param (M)	Val PPL
Original	**8.3**	32.0
Distilled	8.3	30.7
Baseline	159	29.0

* Baseline is Rae et al 2018

Pruning


Pruning
Trained Small Student Model
Pruned Small Student Model

Pruning

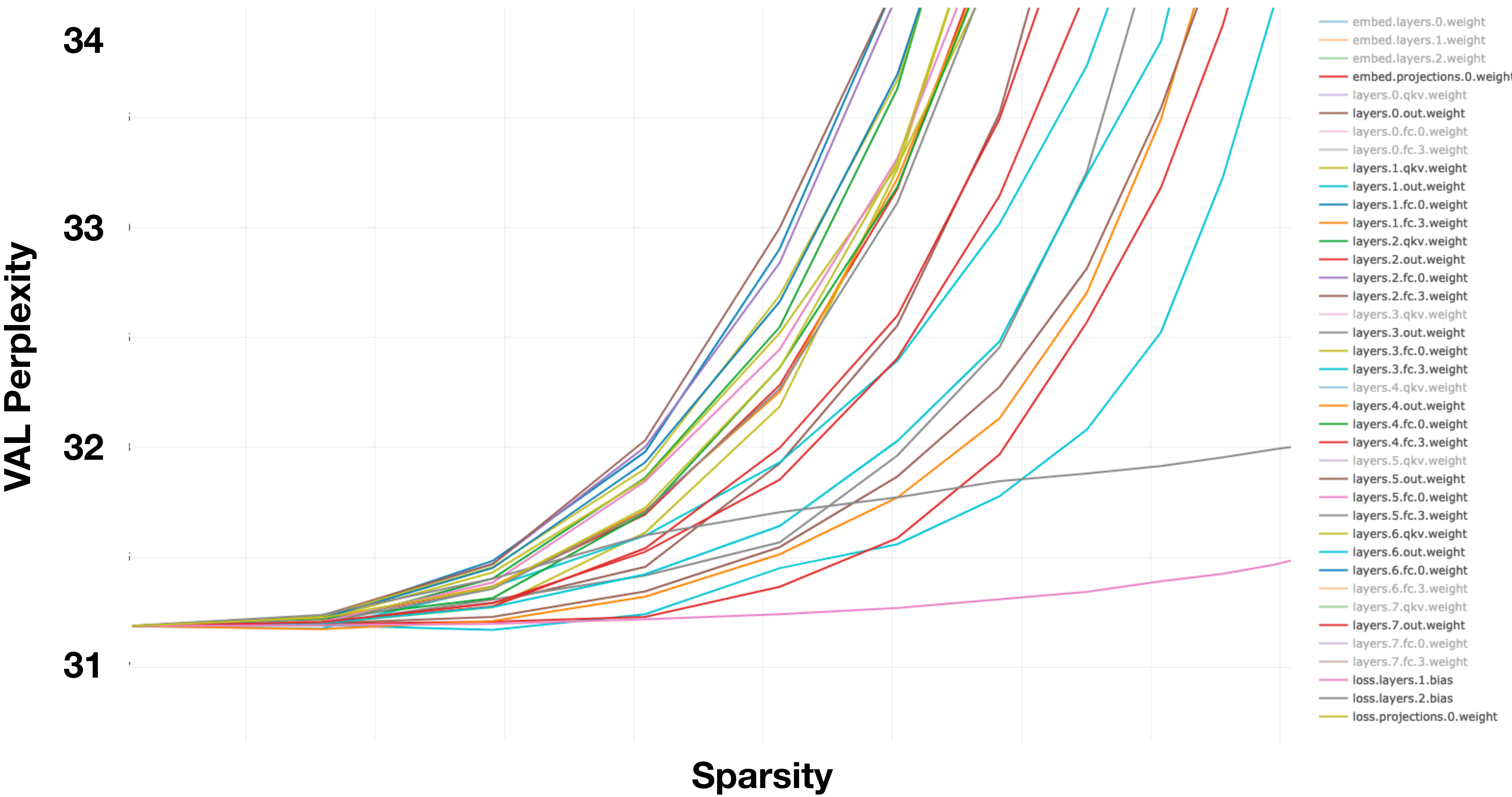


Trained Small Student Model

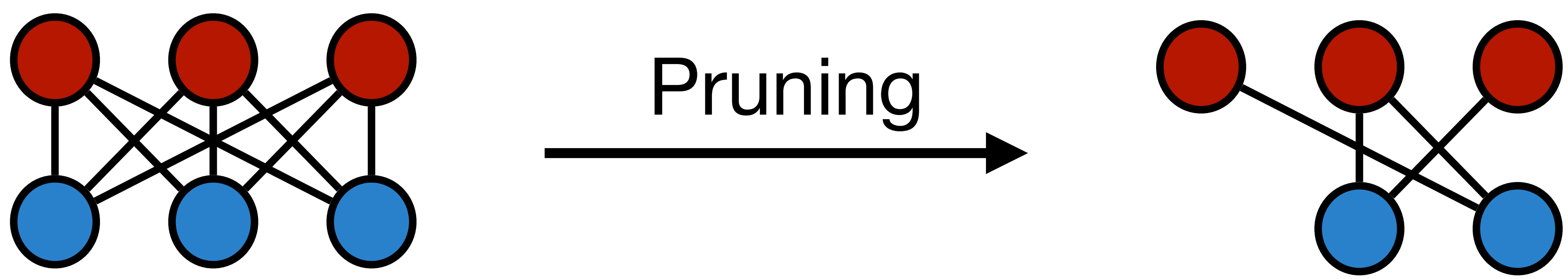
Pruned Small Student Model

- Sensitivity Check

Sensitivity Check



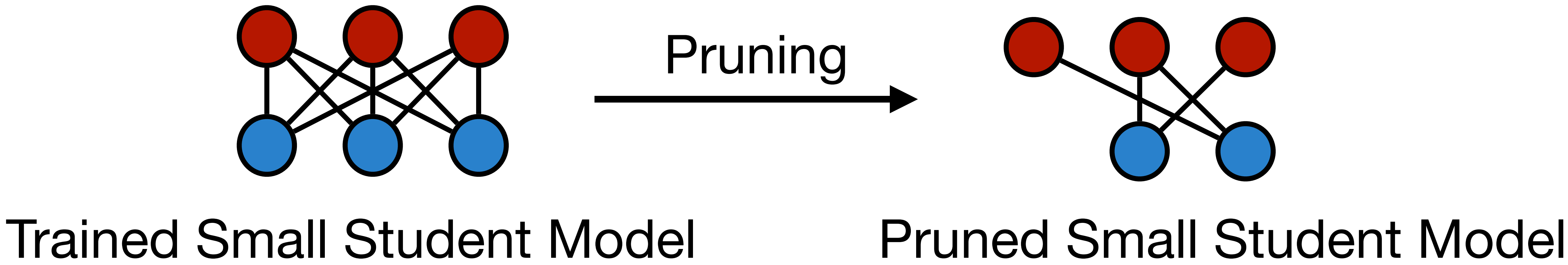
Pruning



Trained Small Student Model　　　Pruned Small Student Model

- Sensitivity Check
- More sensitive layers will be pruned less

Pruning



- Sensitivity Check
- More sensitive layers will be pruned less
- Automatic gradual pruning

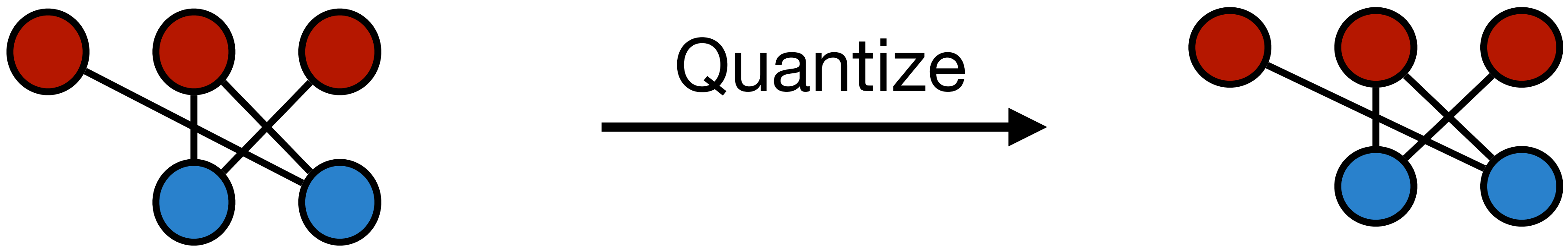
Performance

Model	Sparsity (%)	Param (M)	Compute (M)	Val PPL
1	**42.1**	**5.06**	**11.8**	34.3
2	40.1	5.22	12.1	**34.0**
3	33.9	5.74	13.1	34.3
Baseline	0	159	318	29.0

* Baseline is Rae et al 2018

Quantization

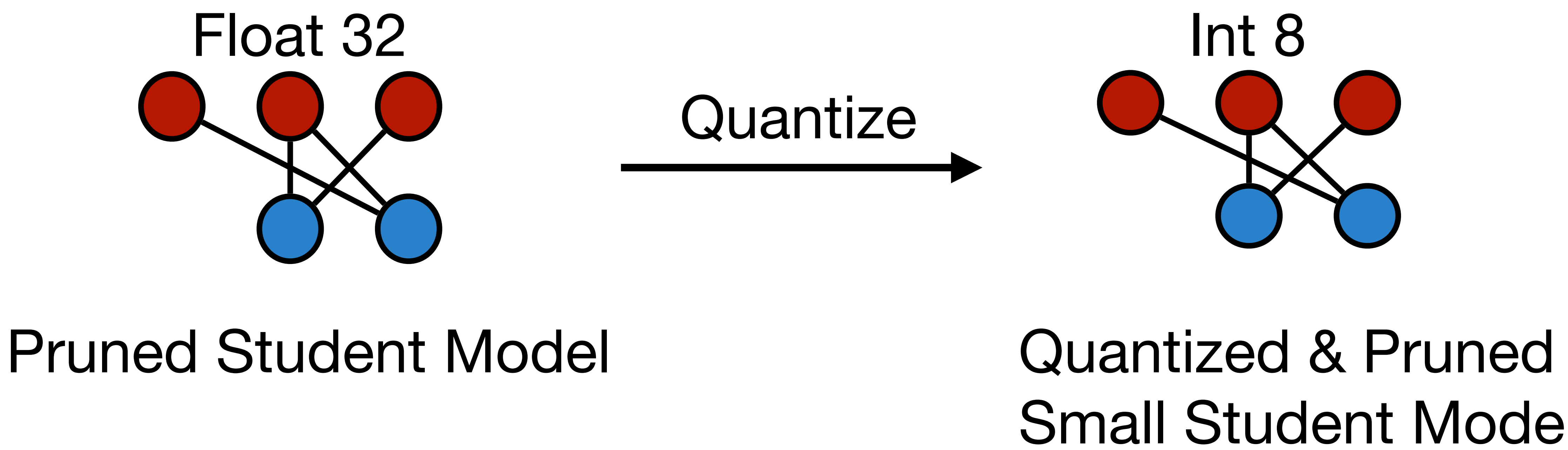
Float 32 Int 8



Pruned Student Model

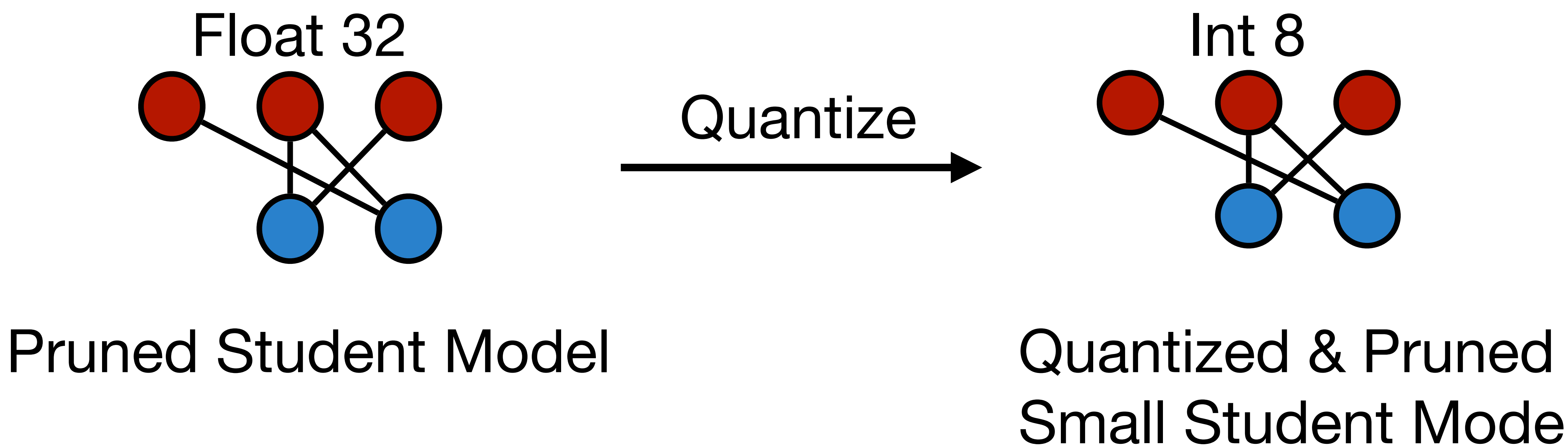
Quantized & Pruned
Small Student Model

Quantization



- Quantize weight, bias and activations of pruned model

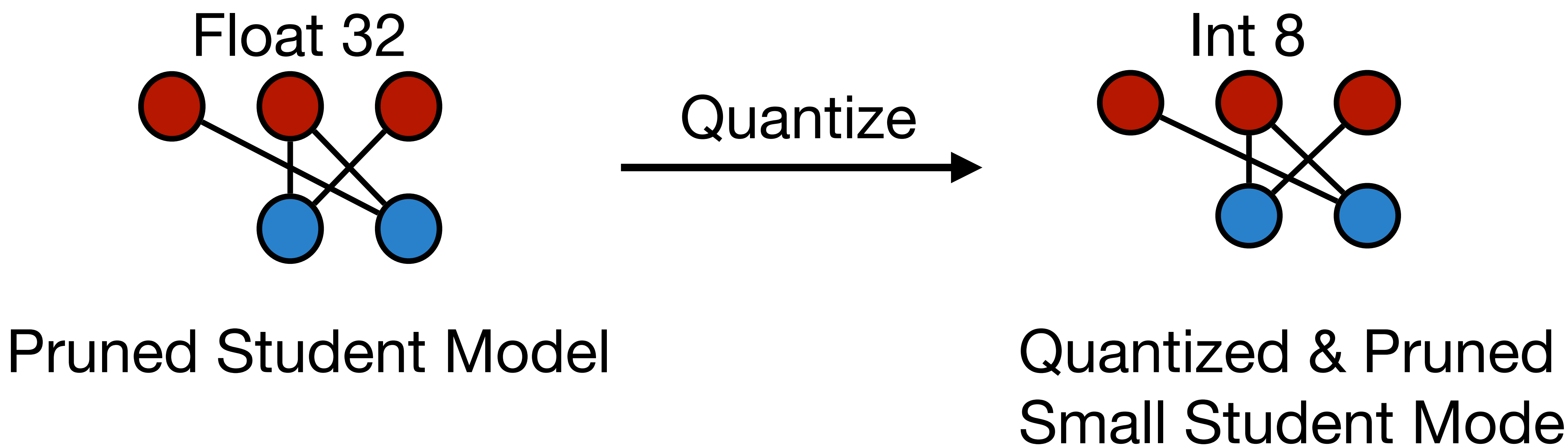
Quantization



- Quantize weight, bias and activations of pruned model

- Linear-range symmetric quantization

Quantization



- Quantize weight, bias and activations of pruned model
- Linear-range symmetric quantization
- Quantize to 8 or 9 bits based on trade-offs between quantization and pruning ratio

Results

Entry	Sparsity (%)	Quantize (bits)	32-bit Param (M)	32-bit Compute (M)	Val PPL	Test PPL	Score
1	**42.1**	9	**1.61**	**7.83**	34.3	34.95	**0.0347**
2	40.1	9	1.65	8.03	**34.0**	**34.7**	0.0356
3	33.9	**8**	1.63	8.48	34.3	34.95	0.0369
Baseline	0	16 (freebie)	~79.5	~239	29.0	29.2	~1.25

Score = param/159M + compute/318M

50x #params reduction and **31x** computation reduction than baseline

* Baseline is Rae et al 2018

Open Source

GitHub open source of our models

Thank you!